SAÉ associée à la ressource R4.03 Qualité et au-delà du relationnel

Ce travail est à effectuer en binôme d'étudiants du même groupe (A ou B ou C) ; si l'effectif de votre groupe est impair, votre enseignant de TD choisira entre un monôme ou un trinôme ou un binôme avec un étudiant d'un autre groupe.

À remettre en déposant, au plus tard le vendredi 10 février 2023 à 23h59, sur https://moodle1.u-bordeaux.fr/mod/assign/view.php?id=603721, un unique fichier archive d'au plus 1 Mio au format `S4◁groupe▷_◁nom 1er étudiant▷_◁nom 2nd étudiant▷.◁extension▷`.

L'évaluation tiendra compte du respect du format du fichier archive déposé, de la qualité de tout votre travail et notamment du code (commentaires, mnémonicité, indentation (grandement facilitée par le langage de programmation imposé), style, respect de normes de programmation, simplicité, efficacité voire complexité optimale, etc.), mais aussi de la pertinence du fichier texte décrivant un ensemble de dépendances fonctionnelles que vous aurez créé.

Vous aurez peut-être à faire une démonstration à votre enseignant de TD.

Vous devez développer une application en programmant en langage Python les fonctions sur la normalisation (dépendances fonctionnelles et formes normales) vues en cours et en TD R4.03 « Qualité et au-delà du relationnel ».

Vous devez donc, a minima, programmer les trois fonctions :

- `FermTransAttr(`$\mathscr{F}$, $\mathscr{X}$ `:` $\mathscr{Y}$
  qui renvoie la fermeture transitive $\mathscr{Y}$ d'un ensemble d'attributs $\mathscr{X}$ pour l'ensemble de dépendances fonctionnelles $\mathscr{F}$
- `CouvMinDF(`$\mathscr{F}$ `:` $\mathscr{C}$
  qui renvoie une couverture minimale $\mathscr{C}$ pour l'ensemble de dépendances fonctionnelles $\mathscr{F}$
- `DecompoDFen3FN(`$\mathscr{F}$, $\mathscr{X}$`:` $\mathscr{S}$
  qui renvoie un schéma relationnel (schémas des relations, contraintes de clé [primaire] voire contraintes d'intégrité référentielles, mais pas les contraintes existentielles, d'unicité ou autres) en 3e forme normale $\mathscr{S}$ en décomposant un ensemble de dépendances fonctionnelles $\mathscr{F}$ et pour un ensemble d'attributs $\mathscr{X}$

Vous devez également programmer la lecture d'un fichier texte décrivant un ensemble de dépendances fonctionnelles $\mathscr{F}$ respectant la syntaxe des arcs du langage `DOT` https://fr.wikipedia.org/wiki/DOT_(langage), $\mathscr{F}$ pouvant ainsi ensuite être géré en mémoire principale.
Quelques exemples de tels fichiers, notamment issus du cours, sont disponibles sur la plateforme pédagogique de l'université de Bordeaux (Sciences & Techno.).

Voici une illustration possible pour le fichier texte `HTS+_simplif.txt` (une version simplifiée de la base de données « HTS+ ») contenant l'ensemble de dépendances fonctionnelles :

```
CLS_Num -> CLS_Intit
ELV_Id -> CLS_Num , ELV_Nom , ELV_AnneeNais , ELV_VilleNais
ELV_Id , RCP_Annee -> RCP_
ELV_Id , EPR_Abrv -> EPR_Note
```

et générant le schéma relationnel en 3e forme normale suivant :

```
Les schémas des relations :
  CLS_Num ( CLS_Num , CLS_Intit )
  ELV_Id ( ELV_Id , ELV_AnneeNais , ELV_Nom , ELV_VilleNais , CLS_Num )
  ELV_Id_RCP_Annee ( ELV_Id , RCP_Annee , RCP_ )
  ELV_Id_EPR_Abrv ( ELV_Id , EPR_Abrv , EPR_Note )
Les contraintes de clé [primaire] :
  L'attribut CLS_Num est la clé de la relation CLS_Num
  L'attribut ELV_Id est la clé de la relation ELV_Id
  Le tuple d'attributs ( ELV_Id , RCP_Annee ) est la clé de la relation
     ELV_Id_RCP_Annee
  Le tuple d'attributs ( ELV_Id , EPR_Abrv ) est la clé de la relation ELV_Id_EPR_Abrv
Les contraintes d'intégrité référentielles :
  La clé étrangère ELV_Id.CLS_Num référence la clé primaire CLS_Num.CLS_Num
  La clé étrangère ELV_Id_RCP_Annee.ELV_Id référence la clé primaire ELV_Id.ELV_Id
  La clé étrangère ELV_Id_EPR_Abrv.ELV_Id référence la clé primaire ELV_Id.ELV_Id
```

Vous devez également créer un fichier texte décrivant un ensemble de dépendances fonctionnelles, de votre choix mais pertinent (c'est à vous de construire un tel jeu d'essai, mettant notamment en évidence les différentes simplifications possibles effectuées lors du calcul d'une couverture minimale de dépendances fonctionnelles voire illustrant les propriétés des dépendances fonctionnelles, et sans reprendre un exercice de cours ou de TD), dont les attributs sont exactement toutes les lettres (sans répétition) des noms des membres de votre binôme ou monôme ou trinôme. Vous trouverez un exemple pour Guibert/Sopena.

Vous devez aussi effectuer autant de tests unitaires https://docs.python.org/3/library/unittest.html que nécessaire pour chacune des trois fonctions `FermTransAttr`, `CouvMinDF` et `DecompoDFen3FN`.

Vous devez finalement générer (automatiquement à partir des commentaires spécifiques) toute la documentation de votre application.

Excepté pour les seuls noms imposés des trois fonctions `FermTransAttr`, `CouvMinDF` et `DecompoDFen3FN`, suivez impérativement les recommandations de la *PEP 8 – Style Guide for Python Code* https://peps.python.org/pep-0008/ (et notamment celles de nommage).
Utilisez également systématiquement `typing` https://docs.python.org/3/library/typing.html.
En revanche, à vous de décider si vous voulez faire de la programmation orientée objet ou non.

Le livrable doit être une archive contenant toute votre application sous la forme d'un premier module Python, tous les tests unitaires sous la forme d'un second module Python, toute la documentation générée, et le fichier texte décrivant l'ensemble de dépendances fonctionnelles pertinent que vous avez créé.